Cuts in Graphs with Matroid Constraints

Aritra Banik
National Institute of Science, Education and Research, An OCC of Homi Bhabha National Institute, Bhubaneswar, Odisha, India

Fedor V. Fomin
University of Bergen, Norway

Petr A. Golovach
University of Bergen, Norway

Tanmay Inamdar
Indian Institute of Technology Jodhpur, India

Satyabrata Jana
University of Warwick, UK

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Norway

Abstract

Vertex (s,t)-Cut and Vertex Multiway Cut are two fundamental graph separation problems in algorithmic graph theory. We study matroidal generalizations of these problems, where in addition to the usual input, we are given a representation $R \in \mathbb{F}^{r\times n}$ of a linear matroid $\mathcal{M} = (V(G), \mathcal{I})$ of rank r in the input, and the goal is to determine whether there exists a vertex subset $S \subseteq V(G)$ that has the required cut properties, as well as is independent in the matroid $\mathcal{M}$. We refer to these problems as Independent Vertex (s,t)-cut, and Independent Multiway Cut, respectively. We show that these problems are fixed-parameter tractable (FPT) when parameterized by the solution size (which can be assumed to be equal to the rank of the matroid $\mathcal{M}$). These results are obtained by exploiting the recent technique of flow augmentation [Kim et al. STOC '22], combined with a dynamic programming algorithm on flow-paths á la [Feige and Mahdian, STOC '06] that maintains a representative family of solutions w.r.t. the given matroid [Marx, TCS '06; Fomin et al., JACM]. As a corollary, we also obtain FPT algorithms for the independent version of Odd Cycle Transversal. Further, our results can be generalized to other variants of the problems, e.g., weighted versions, or edge-deletion versions.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Matroids and greedoids; Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases s-t-cut, multiway Cut, matroid, odd cycle transversal, feedback vertex set, fixed-parameter tractability

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.17

Related Version *Full Version*: `https://arxiv.org/abs/2406.19134` [2]

Funding *Fedor V. Fomin*: Supported by the Research Council of Norway via the project BWCA (grant no. 314528).
Petr A. Golovach: Supported by the Research Council of Norway via the project BWCA (grant no. 314528).
Satyabrata Jana: Supported by the Engineering and Physical Sciences Research Council (EPSRC) via the project MULTIPROCESS (grant no. EP/V044621/1).
Saket Saurabh: The author is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416); and he also acknowledges the support of Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

1 Introduction

Studying problems in graphs and sets with additional constraints given by algebraic structures such as the matroid or submodularity is one of the well-established research directions. These versions not only generalize classical problems, but also show combinatorial interplay between graph structure and linear algebra. Some of the well-known problems in this direction include subsmodular versions of Vertex Cover, Shortest Path, Matching, or Min-(s,t)-Cut [10, 13] as well as coverage problems such as Max Coverage with matroid constraints [3, 30]. In this paper we study some of the classical cut problems in the realm of parameterized complexity in the presence of constraints given by matroids.

It is customary to begin the discussion on our framework with Vertex Cover (given a graph G and a positive integer k, find a subset of size at most k that covers all edges), which is a poster-child problem in parameterized complexity. Among its myriad of generalizations studied in the literature, one of particular relevance to the present discussion is ranked version of Vertex Cover arising in the works of Lovász [19, 20]. He defined Rank Vertex Cover in his study of critical graphs having k-sized vertex cover, by defining the notion of a *framework*, also known as *pregeometric graphs*, which refers to a pair $(G, \mathcal{M})$, where G is a graph and $\mathcal{M} = (V(G), \mathcal{I})$ is a matroid on the vertex set of G. In Rank Vertex Cover, we are given a framework $(G, \mathcal{M})$ and a parameter k, and the goal is to determine whether G has a vertex cover S whose rank – which is defined as the maximum-size independent subset of S – is upper bounded by k. An FPT algorithm for Rank Vertex Cover (or indeed Rank d-Hitting Set) parameterized by the rank is folklore. Meesum et al. [26] used Rank Vertex Cover as a natural target problem for giving a compression from a different parameterization for Vertex Cover. However, even for a slightly more involved problem, such as Feedback Vertex Set (FVS), the ranked version immediately becomes W[1]-hard parameterized by the rank. This is similar to the phenomena seen when we consider weighted problems such as Weighted Vertex Cover parameterized by weight [27, 29].

Having faced with such a obstacle, we study an alternate formulation of FVS in frameworks. In Independent FVS (IFVS), where we are given a framework $(G, \mathcal{M})$, we want to determine whether there exists some $S \subseteq V(G)$ such that (i) S is an independent set in the matroid, and (ii) $G - S$ is acyclic. This happens to be a "sweet spot" between the vanilla version and the ranked version – we can adopt the textbook algorithms, and obtain a polynomial kernel and $\mathcal{O}^*(c^k)$ time FPT algorithm when parameterized by k ($\mathcal{O}^*(\cdot)$ suppresses polynomial factors in the input size.), which is equal to the solution-size (and can also be assumed to be equal to the rank of the matroid, by appropriately truncating it). This was the starting point of our result. Note that IFVS reduces to vanilla FVS when the matroid $\mathcal{M}$ is a uniform matroid of rank k, and hence is NP-hard. Indeed, this gives us a recipe for defining the "independent" versions of classical vertex-subset problems in graphs. We give a formal definition of a prototypical problem below.

Independent Π (where Π is a vertex-subset problem)

Input: A graph $G = (V, E)$, a matroid $\mathcal{M} = (V, \mathcal{I})$ of rank r.
Question: Does there exist a set $S \subseteq V(G)$ such that

- S is an independent set in $\mathcal{M}$, i.e., $S \in \mathcal{I}$, and
- S is a feasible solution for Π

On the other hand, even for a classically polynomial-time solvable problem such as Vertex (s,t)-cut, the corresponding independent version can be shown to be NP-hard even when $\mathcal{M}$ is a partition matroid [12, 21].

1.1 Our Results and Methods

Note that (Vertex) (s,t)-cut forms the bedrock of classical graph theory and algorithms via Menger's theorem and cut-flow duality. Thus, numerous constrained versions of (s,t)-cut have been studied in the literature, and matroid independence is a fundamental constraint that remains unexplored to the best of our knowledge. Another motivation for studying Independent Vertex (s,t)-cut (IVstC) comes from the fact that it appears as a natural subproblem while trying to solve the independent version of Odd Cycle Transversal (aka OCT – given G, k, check whether we can delete at most k vertices from G such that the remaining graph is bipartite) via the so-called iterative compression technique [28]. Thus, we would like to design an FPT algorithm for IVstC, and in turn IOCT, with running times that are as close to the running respective vanilla versions (note that (s,t)-cut is in P, whereas OCT admits an $\mathcal{O}^*(3^k)$ time algorithm, see [28]).

There are two possible approaches that one could take for designing an FPT algorithm for a constrained (s,t)-cut problem, such as IVstC. First approach is to reduce the treewidth of the input graph by using the methodology of Marx, O'Sullivan, and Razgon [24, 25], and then perform a dynamic programming on a bounded treewidth graph. While designing a dynamic programming algorithm for IVstC on bounded-treewidth graphs appears to be straightforward using the technique of representative sets ([23, 9]), it is not obvious that we could use this result as a black-box–we have to be careful while performing any modifications to the graph, since we also have a matroid on the vertex-set. Further, since the treewidth bound obtained via this result is exponential in the size of the solution we are trying to find, it seems that such an approach would result in a double-exponential running time. Thus, we do not pursue this approach further, and the following question remains.

Question 1. Does Independent Vertex (s,t)-cut (and in turn, Independent Odd Cycle Transversal) admit a single-exponential, preferably a $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time FPT algorithm?

Another promising approach to tackle this question is given by Feige and Mahdian [7], who give a very general recipe for solving constrained minimum (s,t)-cut problems. This is done via performing dynamic programming on the flow-paths obtained via Menger's theorem. Naturally, in order to use this approach, it is crucial that we are looking for a *minimum* (s,t)-cut with certain additional constraints. However, this appears to be a roadblock since a *minimum independent* (s,t)-cut can be much larger than the minimum (not-necessarily-independent) (s,t)-cut, and hence Menger's theorem is not applicable. This is where the recent technique of flow augmentation [14, 16] comes to the rescue. At a high level, this a powerful tool that can "augment" the graph by adding additional edges, thus lifting the size of minimum (s,t)-cut to a higher value (hence, *flow augmentation*), such that an unknown *minimal* (s,t)-cut becomes a *minimum* (s,t)-cut. Note that minimality of a solution can be assumed w.l.o.g. since it is compatible with independence as well as cut property. Indeed, all of the aforementioned problems, namely, FVS, OCT, (s,t)-cut satisfy the minimality condition. Hence, using flow augmentation, at the overhead of $2^{\mathcal{O}(k^4 \log k)} \cdot n^{\mathcal{O}(1)}$, we can assume that we are looking for an independent set that is a minimum (s,t)-cut in the augmented graph. However, due to certain technical obstacles while applying flow augmentation, we need to solve the *directed version* of (s,t)-cut using the approach of [7]. Now, we perform a dynamic programming on the flow-paths given by a decomposition theorem from [7] (which needs to be extended to directed graphs), and maintain a representative family of all partial solutions in the DP table. Note that for this, we assume that the matroid $\mathcal{M}$ is given as a representation matrix $R \in \mathbb{F}^{r \times n}$, and the field operations can be performed

Table 1 Our Results.

Π in Independent Π	Matroid access	Result
Feedback Vertex Set	Oracle	$\mathcal{O}^*(10^k)$ and poly kernel
Vertex/Edge (s,t)-cut	Representation	$\mathcal{O}^*(2^{\mathcal{O}(k^4 \log k)})$
Vertex (s,t)-Cut	Oracle	No $f(k) \cdot n^{o(k)}$ algorithm
Odd Cycle Transversal	Representation	$\mathcal{O}^*(2^{\mathcal{O}(k^4 \log k)})$
Multiway Cut	Representation	$\mathcal{O}^*(2^{\mathcal{O}(k^4 \log k)})$

in polynomial-time [1]. Thus, in fact, our algorithm solves a more general problem where it outputs the representative family of all minimum independent (s,t)-cuts. This property is crucial when we want to use this problem as a subroutine to solve a more general problem, to which we return in the next paragraph. Furthermore, due to versatility of the representative sets framework, our algorithms also extend to the weighted versions of these problems, where we want to find a minimum-weight independent set that is a feasible solution to Π. Note that one of the original motivations for the flow augmentation technique was solving one such problem, i.e., Bi-Objective Directed (or weighted) (s,t)-cut [16]. Next, having (almost) answered **Question 1**, we look toward an even more general problem.

Independent Multiway Cut. (Vertex) Multiway Cut (MWC) is a natural generalization of (s,t)-cut, where we are given a set $T \subseteq V(G)$ of *terminals*, and the goal is to find $S \subseteq V(G) \setminus T$ such that $G - S$ disconnects all pairs of terminals from each other. This been studied extensively in the domain of parameterized and approximation algorithms. Note that the vanilla version of the problem was classically shown to be FPT using the technique of important separators [22]. On the other hand, it is not clear whether this technique is any useful in presence of matroid constraints, i.e., for Independent Multiway Cut (IMWC). Even when armed with an FPT algorithm for Independent (s,t)-cut as a subroutine, it is not trivial to design an FPT algorithm for IMWC.

The algorithm initially begins by trying to find a multiway cut (separator) S of size at most k for the terminal set T in G – here we completely ignore independence constraints, and such a separator can be found in $\mathcal{O}^*(2^k)$ time using the result of Cygan et al. [6]. Note that if the graph lacks such a separator, it also cannot have an independent separator. W.l.o.g. suppose that we find a separator S that is also minimal. Hence, for every vertex v in the set S, there exists a path that connects a pair of terminal vertices and intersecting the set S solely at v. This fact brings us to the conclusion that every independent separator either includes v or some vertices of terminal components (we say that a connected component in $G - S$ is a *terminal component* if it contains some terminal). Moreover, each vertex of S has a neighbor in at least two terminal components. With this information, we attempt to design a recursive algorithm by assuming that the "true solution" O to IMC of size k either intersects S, or some terminal component (that can be identified). However, it is also possible that a terminal component contains the entire solution O. In this case, we cannot make a recursive call to the same algorithm, since the parameter does not decrease. To address this scenario,

[1] A matroid $\mathcal{M}$ is representable over a field $\mathbb{F}$ if there exists a matrix $R \in \mathbb{F}^{r \times n}$, and a bijection between the ground set of $\mathcal{M}$ and the columns of R, such that a set is independent in $\mathcal{M}$ iff the corresponding set of columns is independent in $\mathbb{F}$.

we introduce the concept of a *strong separator* (Definition 15), which is a minimal separator S with the following properties: any minimal solution O satisfies (i) $O \cap S \neq \emptyset$, or (ii) there exist at most $k+1$ terminal components – that can be identified in polynomial time – such that O intersects at least one of them, with the size of intersection being between 1 and $k-1$. Further, such a strong separator can be found in time $\mathcal{O}^*(2^k)$ time (Lemma 16) by applying "pushing arguments". This lets us design a recursive divide-and-conquer algorithm, and due to property (ii), we know that in each recursive call, the parameter always strictly decreases. For handling the matroid independence constraints, in fact we need to strengthen the algorithm to return a representative family of all possible multiway cuts. This is where it is useful that our algorithm for IVstC also returns such a family, since this algorithm is used in the base case of the algorithm (when $|T| = 2$). This strategy results $\mathcal{O}^*(2^{\mathcal{O}(k^4 \log k)})$ time algorithm for IMWC, where the running time is dominated by the flow augmentation step required in the base case of IVstC.

Due to lack of space, we omit some of the details from the description of IVstC algorithm (Section 3), and we give a very brief sketch of the algortihm for IMWC (Section 4). The other results from Table 1 are omitted and can be found in the full version [2].

Motivation for independence constraints. One motivation for studying such independent versions of classical graph problems comes from the fact that matroid-independence captures several interesting constraints on the solution vertices. One prominent example of this is the *colorful* versions of the problem that arise naturally from the *color-coding* technique introduced by Alon, Yuster and Zwick [1], where a problem such as k-Path, i.e., determining whether the given graph contains a path of length at least k. While k-Path itself is difficult to approach using combinatorial techniques, there is a simple dynamic programming algorihtm, if we want to find a colorful k-path, i.e., where we are given a graph whose vertices are colored from $\{1, 2, \ldots, k\}$, and we want to determine whether the graph contains a path of length k whose vertices are colored with distinct colors [1]. Then, a standard argument shows that a hypothetical unknown solution will get colored with distinct colors if each vertex is assigned a color uniformly at random. On the flip side, colorful versions of problems such as Independent Set, Clique, are also convenient starting points in various W[1]-hardness reductions. Note that colorful constraints is one of the simplest examples of matroid constraints – it is simply a partition matroid, where the coloring of vertices corresponds to a partition of the vertex set, and the solution may contain (at most) one vertex from each partition. Slightly more general partition matroid constriants can capture "local budgets" on the solution from each class of vertices, which can be used to model different fairness constraints, i.e., we cannot delete too many vertices of the same type to achieve certain graph property, which can be "unfair" for that type. Such constraints on solution are rather popular in clustering [11, 17], which are indeed motivated from such fairness considerations.

2 Preliminaries

Notations. We use standard graph theory notation. Here we only give some crucial definitions and refer the reader to the preliminary section in the full version [2]. For a graph G and two vertices $s, t, \in V(G)$, we say that vertex set $S \subseteq V(G)$ is an (s,t)-cut if there is no path from s to t in $G - S$. For a set $T \subseteq V(G)$ of *terminal vertices*, we say that S is a *multiway cut* for T in G, if T is an (t_i, t_j)-cut for every pair of vertices $t_i, t_j \in T$.

Matroids. A pair $\mathcal{M} = (U, \mathcal{I})$ with a finite universe U and a family $\mathcal{I}$ of sets over U that satisfies the following three properties: (i) $\emptyset \in \mathcal{I}$, (ii) if $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$, (iii) if $A \in \mathcal{I}$ and $B \in \mathcal{I}$ and $|A| < |B|$ then there is an element $b \in B \setminus A$ such that $A \cup \{b\} \in \mathcal{I}$. Each set $S \in \mathcal{I}$ is called an independent set. The size of maximal independent set in $\mathcal{M}$ is called its rank, and is denoted by $r = \mathsf{rank}(M)$. We will use $U(\mathcal{M})$ and $\mathcal{I}(\mathcal{M})$ to refer to the ground set and the independent-set collection of $\mathcal{M}$, respectively.

A matroid $\mathcal{M} = (U, \mathcal{I})$ is said to be *representable* over a field $\mathbb{F}$ if there exists a matrix $R \in \mathbb{F}^{r \times n}$ where $r = \mathsf{rank}(\mathcal{M}), |U| = n$, and a bijection from U to the columns of R, such that for any $S \subseteq U$, $S \in \mathcal{I}$ iff the corresponding set of columns is linearly independent over $\mathbb{F}$. Contraction and truncation are two natural operations defined on matroids, whose definition can be found in the full preliminaries [2].

We call a family of subsets (of U) of size $p \geq 0$ as a *p-family*. Next, we state the following crucial definition of representative families.

▶ **Definition 1** ([8, 4]). *Let $\mathcal{M} = (U, \mathcal{I})$ be a matroid. For two subsets $A, B \subseteq U$, we say that A* fits *B (or equivalently B* fits *A), if $A \cap B = \emptyset$ and $A \cup B \in \mathcal{I}$.*

Let $\mathcal{A} \subseteq \mathcal{I}$ be a p-family for some $0 \leq p \leq \mathsf{rank}(\mathcal{M})$. A subfamily $\mathcal{A}' \subseteq \mathcal{A}$ is said to q-represent *$\mathcal{A}$ if for every set B of size q such that there is an $A \in \mathcal{A}$ that fits B, then there is an $A' \in \mathcal{A}'$ that also fits B. If $\mathcal{A}'$ q-represents $\mathcal{A}$, we write $\mathcal{A}' \subseteq^q_{rep} \mathcal{A}$.*

The following proposition lets us compute represnetative families efficiently.

▶ **Proposition 2** ([8, 4]). *There is an algorithm that, given a matrix R over a field $\mathbb{F}$, representing a matroid $\mathcal{M} = (U, \mathcal{I})$ of rank k, a p-family $\mathcal{A}$ of independent sets in $\mathcal{M}$, and an integer q such that $p + q = k$, computes a q-representative family $\mathcal{A}' \subseteq^q_{rep} \mathcal{A}$ of size at most $\binom{p+q}{p}$ using at most $\mathcal{O}(|\mathcal{A}|(\binom{p+q}{p})p^\omega + (\binom{p+q}{p})^{\omega-1})$ operations over $\mathbb{F}$. Furthermore, an analogous result holds even if $p + q \leq k$, albeit depending on the field $\mathbb{F}$, the algorithm may be randomized.* [2]

For a p-family $\mathcal{P}$ and a q-family $\mathcal{Q}$, we define $\mathcal{P} \bullet \mathcal{Q} := \{P \cup Q : P \in \mathcal{P}, Q \in \mathcal{Q}, P \text{ fits } Q\}$ (this operation is also known as subset convolution). The following properties of representative families follow straightfordly from the definitions, and formal proofs can be found in [5].

▶ **Proposition 3** ([5]). *Let $\mathcal{M} = (U, \mathcal{I})$ be a matroid and $\mathcal{A}$ be a p-family.*

- *If $\mathcal{A}_1 \subseteq^{r-p}_{rep} \mathcal{A}$, and $\mathcal{A}_2 \subseteq^{r-p}_{rep} \mathcal{A}_1$, then, $\mathcal{A}_2 \subseteq^{r-p}_{rep} \mathcal{A}$.*
- *If $\mathcal{A}_1 \subseteq^{r-p}_{rep} \mathcal{A}$, then $\mathcal{A} = \emptyset$ iff $\mathcal{A}_1 = \emptyset$.*
- *Let $\mathcal{P}$ be a p-family and $\mathcal{Q}$ be a q-family. Let $\mathcal{P}' \subseteq^{r-p}_{rep} \mathcal{P}$, and $\mathcal{Q}' \subseteq^{r-q}_{rep} \mathcal{Q}$. Then, $\mathcal{P}' \bullet \mathcal{Q}' \subseteq^{r-p-q}_{rep} \mathcal{P} \bullet \mathcal{Q}$.*

3 FPT Algorithm for Independent Vertex (s,t)-Cut

In this section, we sketch an FPT algorithm for Independent Vertex (s,t)-Cut, and the full description can be found in Section 3 of the full version [2]. To this end, we define some notation. Consider a graph $G = (V, E)$, a matroid $\mathcal{M} = (U, \mathcal{I})$ of rank r where $V(G) \setminus \{s, t\} \subseteq U$, and two vertices $s, t \in V(G)$. By iteratively checking values $k = 1, 2, \ldots,$ and running the algorithm described below, we may inductively assume that we are working

[2] Here, we need to compute the representation of an appropriately truncated matroid. For this, only randomized polynomial algorithm is known in general [23]; however for certain kinds of fields, it can be done in polynomial time [18].

with a value $0 \leq k \leq r$, such that there is no independent (s,t)-cut in G for any $k' < k$. This follows from the fact that, our generalized problem, as defined below, requires the algorithm to return a set corresponding to k' whose emptiness is equivalent to concluding that the there is no independent (s,t)-cut of size k'. It is important to note that, depending on whether we are addressing Independent Vertex (s,t)-Cut as a standalone problem or as a sub-procedure, we may either obtain the smallest value of k for free, or we may need to perform iterations. Regardless, we can safely assume that we are operating with a value of this nature. For any $0 \leq k \leq r$, let

$$\mathcal{F}(s,t,k) \coloneqq \{S \subseteq V(G) \setminus \{s,t\} : S \in \mathcal{I}, |S| = k, S \text{ is a minimal } (s,t)\text{-cut.}\}$$

In fact, we design an FPT algorithm for the following generalization of IVstC.

Generalized Independent Vertex (s,t)-Cut

Input: A graph G, a vertex set $Q \subseteq V(G)$ containing s,t, a representation $R \in \mathbb{F}^{r \times n}$ of a matroid $\mathcal{M} = (U, \mathcal{I})$ of rank r where $V \setminus Q \subseteq U$, an integer $0 \leq k \leq r$.
Output: $\mathcal{F}'(s,t,k) \subseteq^{r-k}_{\text{rep}} \mathcal{F}(s,t,k)$.

Note that this generalizes Independent Vertex (s,t)-Cut, since the latter problem is equivalent to determining whether $\mathcal{F}(s,t,k)$ (or, equivalently, $\mathcal{F}'(s,t,k) \subseteq^{r-k}_{\text{rep}} \mathcal{F}(s,t,k)$, via Proposition 3) is non-empty for some $0 \leq k \leq r$.

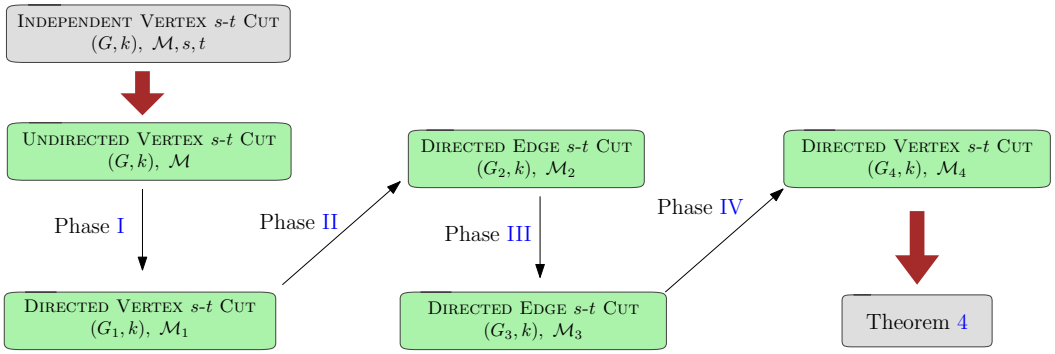

Figure 1 Schematic depiction of sequence of reductions from undirected vertex (s,t)-cut to directed vertex (s,t)-cut via flow augmentation. These details are omitted from the short version.

▶ **Theorem 4.** *Consider an instance $(G, \mathcal{M}, s, t, Q, k)$ of* Generalized Independent Vertex (s,t)-Cut*, where there exists no independent (s,t)-cut in G of size less than k. Then, there exists a deterministic algorithm for* Generalized Independent Vertex (s,t)-Cut *that runs in time $2^{\mathcal{O}(k^4 \log k)} \cdot 2^{\mathcal{O}(r)} \cdot n^{\mathcal{O}(1)}$, and outputs $\mathcal{F}'(s,t,k) \subseteq^{r-k}_{rep} \mathcal{F}(s,t,k)$ such that $|\mathcal{F}'(s,t,k)| \leq 2^r$. Here, $\mathcal{F}(s,t,k)$ is the collection of all independent (s,t)-cuts of size k.*

Flow-augmentation

Below we present the recent result on the (deterministic version of) *flow-augmentation* technique [15, 16], which we will crucially use in our algorithms.

▶ **Proposition 5** ((Deterministic) Flow-augmentation for Directed Graphs [14, 16]). *There exists an algorithm that, given a directed graph G, two vertices $s, t \in V(G)$, and an integer k, in time $2^{\mathcal{O}(k^4 \log k)} \cdot |V(G)|^{\mathcal{O}(1)}$ outputs a set $\mathcal{A} \subseteq 2^{V(G) \times V(G)}$ of size $2^{\mathcal{O}(k^4 \log k)} \cdot (\log n)^{\mathcal{O}(k^3)}$ such that the following holds: for every minimal (s,t)-cut $Z \subseteq E(G)$ of size at most k, there exists $A \in \mathcal{A}$ such that the edge set Z remains an (s,t)-cut in $G + A$ and, furthermore, Z is a minimum (s,t)-cut in $G + A$.*

Algorithm For Directed Vertex (s,t)-cut

Due to a series of reductions via flow augmentation, our task boils down to solving the generalized version of Independent Directed Vertex (s,t)-cut. However, flow augmentation additionally lets us assume that k is equal to the size of the *minimum* (s,t)-cut. We give a formal definition below. Let $k \in [0, r]$ denotes the size of *minimum vertex (s,t)-cut* in G. Then, we define

$$\mathcal{F}(s,t,k) \coloneqq \{X \subseteq V(G) \setminus \{s,t\} : X \text{ is an independent } (s,t)\text{-cut of size } k\}.$$

Here is a formal description of the problem we consider.

Generalized Independent Directed Vertex (s,t)-cut (GIDVC)

Input: A directed graph G, a linear matroid $\mathcal{M} = (U, \mathcal{I})$, a set $Q \subseteq V(G)$ containing s, t, such that $V(G) \setminus Q \subseteq U$, and two non-negative integers k, q such that $k + q \leq r = \mathsf{rank}(\mathcal{M})$, and k is the size of minimum vertex (s,t)-cut.
Output: Return $\mathcal{F}'(s,t,k) \subseteq^q_{\text{rep}} \mathcal{F}(s,t,k)$ of bounded size.

In this section, we adopt the approach of Feige and Mahdian [7] to our end. Many of the definitions are directly borrowed from the same (albeit for directed graphs). Due to space considerations, some proofs are omitted from the short version, and can be found in the full version [2].

▶ **Definition 6** (Critical and non-critical vertex). A vertex v of G is called *critical* if every collection of k vertex-disjoint paths from s to t contains v. A vertex is *non-critical* if it is not critical.

▶ **Proposition 7.** *A vertex v is critical if and only if there is a directed vertex (s,t)-cut of size k containing v.*

▶ **Definition 8** (Connecting pairs). For a pair of vertices u and v in G, we say that u is connected to v in G if there is a directed path from u to v such that every internal vertex of the path is non-critical.

Fix a collection of k vertex-disjoint paths $P_1, P_2, \ldots, P_k$ from s to t. By definition, each critical vertex must be on one of these paths. In addition, Proposition 7 implies the following.

▶ **Observation 9.** *For every $S \in \mathcal{F}(s,t,k)$, all $v \in S$ are critical vertices. Furthermore, distinct vertices of S belong to distinct paths from $\{P_1, P_2, \ldots, P_k\}$.*

For each P_i , let $v_{i,1}, \ldots, v_{i,z_i}$ be the sequence of critical vertices of P_i in the order in which they appear on this path from s to t. Here, z_i denotes the number of critical vertices in P_i. To simplify notation, we define $v_{i,0} = s$ and $v_{i,z_i+1} = t$ for every i, and treat the vertices s and t as critical. Note that in the problem definition, we are not allowed to be chosen as

solution vertices (i.e., cut vertices). Let Ω be the set of all k-tuples $\boldsymbol{a} = (a_1, \ldots, a_k)$ where $1 \leq a_i \leq z_i$ for every i. In other words, each $\boldsymbol{a} \in \Omega$ corresponds to one way of selecting a critical vertex from each P_i. Note that this does not have to correspond to an (s,t)-cut in G, since there are edges and vertices in G that are not in the paths P_i's.

▶ **Definition 10** (Prefix subgraph). The prefix subgraph $G[\boldsymbol{a}]$ defined by $a \in \Omega$ is an induced subgraph of G with the vertex set defined as follows:

- a critical vertex $v_{i,j}$ is in $G[\boldsymbol{a}]$ if and only if $j \leq a_i$;
- a non-critical vertex u is in $G[\boldsymbol{a}]$ if and only if all critical vertices that are connected to u are in $G[\boldsymbol{a}]$.

The last two layers of $G[\boldsymbol{a}]$ are the set of critical vertices $v_{i,j}$ such that $a_{i-1} \leq j \leq a_i$.

The proof of the following lemma follows the undirected version from [7] and hence omitted from the short version.

▶ **Lemma 11** (Decomposition lemma, [7]). *There is a sequence $\boldsymbol{a}^1, \ldots, \boldsymbol{a}^p \in \Omega$ such that*

(a) *$\boldsymbol{a}^1 = (1, \ldots, 1)$ and $\boldsymbol{a}^p = (z_1, \ldots, z_k)$;*

(b) *for every $h = 2, \ldots, p$, $\boldsymbol{a}^h - \boldsymbol{a}^{h-1}$ is a vector with exactly one entry equal to one, and zero elsewhere; and*

(c) *for every $h = 1, \ldots, p$, every critical vertex of $G[\boldsymbol{a}^h]$, except possibly the vertices in the last two layers of $G[\boldsymbol{a}^h]$, is not connected to any critical vertex not in $G[\boldsymbol{a}^h]$.*

Next, we define the notion of a valid coloring that essentially captures partial solutions that will be used to perform dynamic programming.

▶ **Definition 12** (Valid coloring). A valid coloring of a prefix subgraph $G[\boldsymbol{a}]$ is a partial coloring of the vertices of $G[\boldsymbol{a}]$ with colors **s**ilver (`s`), **t**an (`t`), and **b**lack (`b`) such that

- (black vertices) for each i, there is at most one vertex of $G[\boldsymbol{a}]$ on P_i that is colored black; furthermore, s and t cannot be colored black;
- (silver and tan critical vertices) for each i and $j \leq a_i$, if there is no $j' \leq a_i$ such that $v_{i,j'}$ is colored black, then $v_{i,j}$ must be colored silver; if there is such a j', then $v_{i,j}$ must be colored silver if $j < j'$ and tan if $j > j'$;
- No silver-colored critical vertex is connected to a tan-colored critical vertex; and
- (coloring for non-critical vertices) every non-critical vertex that is connected from at least one silver critical vertex is colored silver; every non-critical vertex that is connected to at least one tan critical vertex is colored tan;
- (some vertex remains uncolored) every non-critical vertex that is neither connected from any silver critical vertex nor connected to any tan critical vertex remains uncolored.

We use col to denote the set of all possible valid colorings of G that color vertex t tan. We use $\mathcal{S}$ to denote all minimum vertex (s,t)-cuts in G.

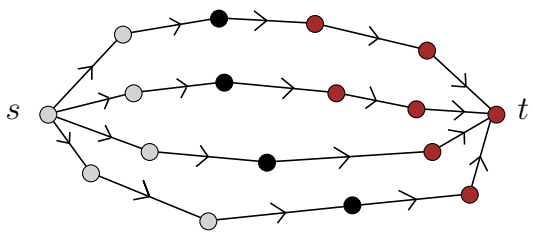

Figure 2 An illustration of a valid coloring on critical vertices.

We start with following lemma, whose proof can be found in the full version [2].

▶ **Lemma 13.** *There is a bijection $f : \mathsf{col} \to \mathcal{S}$, i.e. between the set of all valid colorings of G that color vertex t tan, and the set of all minimum vertex (s,t)-cuts in G.*

Finally, we are ready to prove our main result of this section.

▶ **Theorem 14.** *There exists a deterministic algorithm for* Generalized Directed Independent Vertex (s,t)-cut *that runs in time $2^{\mathcal{O}(r)}$ time, and returns $\mathcal{F}'(s,t,k) \subseteq^{r-k}_{rep} \mathcal{F}(s,t,k)$ of size at most 2^r. Here, k denotes the size of the minimum (s,t)-cut and r denotes the rank of the matroid $\mathcal{M}$.*

Proof. Note that by Lemma 13 any valid coloring of the graph G that colors the vertex t tan ($\mathtt{t}$) corresponds to a (s,t)-cut in G. Conversely, any rank (s,t)-cut in G gives rise to a valid coloring of G that colors t tan. Recall that our goal is to efficiently compute $\mathcal{F}'(s,t,k) \subseteq^{r-k}_{\text{rep}} \mathcal{F}(s,t,k)$ of bounded size. By Lemma 13, there is a bijection between set of all minimum vertex (s,t)-cuts in G and valid colorings with t is colored tan. Therefore, we design a dynamic programming-based algorithm along the decomposition lemma (Lemma 11) to compute "partial representative family" that represents (partial) valid coloring. We describe this DP below.

We use Lemma 11 to construct the sequence $\boldsymbol{a}^1, \ldots, \boldsymbol{a}^p$ in polynomial time. Based on this sequence, we define a table A of dimension $p \times 3^{2k} \times k$ A as follows.

- The entry $A[h,y,i]$ is indexed an integer $h \in [p]$, a string $y \in \{\mathtt{s},\mathtt{t},\mathtt{b}\}^{2k}$, and an integer $i \in [k]$.
- Fix some $h \in [p], y \in \{\mathtt{s},\mathtt{t},\mathtt{b}\}^{2k}, i \in [k]$. We say that a valid coloring $\chi : V(G[\boldsymbol{a}^h]) \to \{\mathtt{s},\mathtt{t},\mathtt{b}\}$ *respects* the tuple (h,y,i) if it colors the vertices in the last two layers of $G[\boldsymbol{a}^h]$ according to y, and $|B_\chi| = i$, where $B_\chi \coloneqq \chi^{-1}(\mathtt{b})$. We define $\mathcal{S}(h,y,i)$ to be the set of all such subsets B_χ over all valid colorings χ of $G[\boldsymbol{a}^h]$ that respect (h,y,i). In the entry $A(h,y,i)$, we will store $\widehat{\mathcal{S}} \subseteq^{r-i}_{\text{rep}} \mathcal{S}(h,y,i)$.

The base entries $A[1,\cdot,\cdot]$ can be computed by inspection. We now give a procedure to compute $A[h,y,i]$, based on the entries $A[h-1,\cdot,\cdot]$. The last two layers of $G[\boldsymbol{a}^h]$ differ from the last two layers of $G[\boldsymbol{a}^{h-1}]$ in that one vertex (say v) was added and one vertex (say u) was removed. Let w denote the vertex which is in the last layer of $G[\boldsymbol{a}^{h-1}]$ but in the second last layer of $G[\boldsymbol{a}^h]$. Note that the vertex w always exists. Technically, if $u = s$, then s may still belong to the last two layers of $G[a^h]$, but the treatment of this case is only simpler than in the case $u \neq s$ and is omitted. We try all three possible colors for u. For each color, we first check if in combination with y it violates the condition for a valid coloring. If 'not, then we do the next step. We compute the color of all the vertices that are in $G[\boldsymbol{a}^h]$ but not in $G[\boldsymbol{a}^{h-1}]$ using the fourth condition in Definition 12. Note that by condition (c) of Lemma 11, the color of any such vertex can be uniquely specified. We perform a case analysis based on $y(v)$ and define a family F that we will eventually store in $A[h,y,i]$. Let y' be a coloring of the vertices in the last two layers of $G[\boldsymbol{a}^{h-1}]$. We say that y and y' are *compatible* with each other, if they agree on coloring of the vertices that are in common in their respective last two layers. We denote this by $y \propto y'$.

(i) $y(v) = \mathtt{b}$ As y is a valid coloring we have $y(u) = y(w) = \mathtt{s}$. In this case, if $y' \propto y$ then we must have $y'(u) = y'(w) = \mathtt{s}$. Therefore, there is a unique coloring in the last two layers in the vertices of $G[\boldsymbol{a}^{h-1}]$ that is compatible with y. Let $F = \{Q \cup \{v\} \colon Q \in A[h-1,y',i-1],\ y' \propto y\}$. So here $|F| = |A[h-1,y',i-1]| \leq 2^r$.

(ii) $y(v) = \mathtt{s}$ As y is a valid coloring, we have $y(u) = y(w) = \mathtt{s}$. In this case, if $y' \propto y$ then we must have $y'(u) = y'(w) = \mathtt{s}$. Therefore, there is a unique coloring in the last two layers in the vertices of $G[\boldsymbol{a}^{h-1}]$ that is compatible with y. Let $F = A[h-1,y',i]$. So, here $|F| = |A[h-1,y',i]| \leq 2^r$.

(iii) $y(v) = \mathtt{t}$ As y is a valid coloring, we have either $y(u) = y(w) = \mathtt{t}$, or $y(u) = \mathtt{b}$, $y(w) = \mathtt{t}$, or $y(u) = \mathtt{s}$, $y(w) = \mathtt{b}$. In this case, if $y' \propto y$ then we must have either $y'(u) = y'(w) = \mathtt{t}$, or $y'(u) = \mathtt{b}$, $y'(w) = \mathtt{t}$, or $y'(u) = \mathtt{s}$, $y'(w) = \mathtt{b}$. Here we distinguish these by two subcases based on $y'(w)$. Recall that $y(w) = y'(w)$.

- If $y'(w) = \mathtt{b}$ then we must have $y'(u) = \mathtt{s}$. So in this case, there is a unique coloring in the last two layers in the vertices of $G[\boldsymbol{a}^{h-1}]$ which is compatible with y. Let $F = A[h-1, y', i]$. So here $|F| = |A[h-1, y', i]| \leq 2^r$.
- If $y'(w) = \mathtt{t}$ then u can take color either black or tan. So there are at most two different colorings in the last two layers in the vertices of $G[\boldsymbol{a}^{h-1}]$ which is compatible with y. Let $F' = \bigcup_{y' \propto y} A[h-1, y', i]$. So here $|F'| = |\bigcup_{y' \propto y} A[h-1, y', i]| \leq 2 \cdot 2^r$. Now we use the tool of *representative sets* (Proposition 2) to find a set $F \subseteq^{r-i}_{\text{rep}} F'$ of size at most 2^r.

Finally, we will let $A[h, y, i] \leftarrow F$. Let $\mathcal{Y}$ denote the set of all strings that color the vertex t tan in $G[\boldsymbol{a}^p]$. Note that $|\mathcal{Y}| \leq 2^k$. Given the table A, one can easily check the existence of a desired (s,t)-cut by checking the entries $A[p, y, k]$ for all strings $y \in \mathcal{Y}$. Formally if $A[p, y, k] \neq \emptyset$ for some string y where $y(t) = t$ then we return YES to our problem on the instance G. For the more general problem, when we seek the representative family, we compute $\mathcal{F}' \subseteq^{r-k}_{\text{rep}} \bigcup_{y \in \mathcal{Y}} A[p, y, k]$ and return $\mathcal{F}'$. ◀

▶ **Theorem 4.** *Consider an instance $(G, \mathcal{M}, s, t, Q, k)$ of* Generalized Independent Vertex (s,t)-Cut, *where there exists no independent (s,t)-cut in G of size less than k. Then, there exists a deterministic algorithm for* Generalized Independent Vertex (s,t)-Cut *that runs in time $2^{\mathcal{O}(k^4 \log k)} \cdot 2^{\mathcal{O}(r)} \cdot n^{\mathcal{O}(1)}$, and outputs $\mathcal{F}'(s,t,k) \subseteq^{r-k}_{rep} \mathcal{F}(s,t,k)$ such that $|\mathcal{F}'(s,t,k)| \leq 2^r$. Here, $\mathcal{F}(s,t,k)$ is the collection of all independent (s,t)-cuts of size k.*

4 Vertex Multiway Cut

Like Independent Vertex (s,t)-Cut, our aim here is also to design an algorithm that solves a more general version of the problem, called Generalized Independent Multiway Cut, or GIMC for short. Like Section 3, we will assume that we are working with a value of k such that for any $k' < k$ there is no independent multiway cut of size k'. Then, let $\mathcal{F}(T, k, G) \coloneqq \{S \subseteq V(G) \setminus T : |S| = k, S \in \mathcal{I}$, and S is a multiway cut for $T\}$. Note that, if there is no independent multiway cut of size k, then $\mathcal{F}(T, k, G) = \emptyset$. We give a formal definition of the problem.

Generalized Independent Multiway Cut(GIMC)

Input: An instance $(G, \mathcal{M}, T, Q, k)$, where:

- $G = (V, E)$ is a graph,
- $T \subseteq V(G)$ is a set of terminals, $Q \subseteq V(G)$ is a set of special vertices, with $T \subseteq Q$,
- Matroid $\mathcal{M} = (U, \mathcal{I})$ of rank r, where $V(G) \setminus Q \subseteq U$,
- $0 \leq k \leq r$

Output: $\mathcal{F}'(T, k, G) \subseteq^{r-k}_{\text{rep}} \mathcal{F}(T, k, G)$.

First, in Section 4.1, we give the crucial definitions of certain kind of separator, called a *strong separator*, for GIMC, which can be computed in FPT time, or we can conclude that we have a NO-instance. Then, we sketch our recursive algorithm for GIMC that is FPT in k.

4.1 Finding a "Strong Separator" for GIMC

In this subsection, we define the notion of a "strong separator" and how it interacts with any minimal multiway cut. Let G be a graph and $T \subseteq V(G)$ be a set of terminals. Throughout the section, we will assume that G is connected. Let $S \subseteq V(G)$ be a minimal multiway cut for T, and let $\mathcal{C}_S$ denote the set of connected components of $G - S$. If a connected component $C \in \mathcal{C}_S$ contains a terminal vertex then we call it a *terminal component*; otherwise, we call it a *non-terminal component*. Let $\mathcal{T}_S$ denote the terminal components and $\mathcal{N}_S$ denote the non-terminal components in $G - S$. Note that $\mathcal{C}_S = \mathcal{T}_S \uplus \mathcal{N}_S$. We say that a $v \in S$ is adjacent to a component $C \in \mathcal{C}_S$ if there exists some $u \in C$ such that uv is an edge. Given two components C_1 and C_2 we say C_1 is adjacent to C_2 (and, vice versa) if there is a vertex in C_1 which has a neighbour in C_2. We will also often refer to a (not-necessarily-independent) multiway cut S as a *separator*. Next we define the notion of a strong separator.

▶ **Definition 15** (Strong separator (modified from the full version [2]))**.** *Given an instance $(G, \mathcal{M}, T, Q, k)$, we say S is* strong separator *for T in G if for any minimal solution O of size k, (i) either $S \cap O \neq \emptyset$, or (ii) there exist $q \leq k+1$ terminal components $C_1, C_2, \ldots, C_q$ in $G - S$ such that $1 \leq |O \cap C_i| \leq k-1$ for some $i \in [q]$.*

In the following crucial result, we show that such a strong separator and the corresponding q components $C_1, C_2, \ldots, C_q$ can be computed in FPT time. Much of Section 4.1 in the full version [2] is dedicated to the proof of this lemma, but the proof is omitted from here.

▶ **Lemma 16.** *There exists an algorithm that takes an input an instance $(G, \mathcal{M}, T, Q, k)$ of* GIMC*, where G is a connected graph on n vertices, $k \geq 2$, and $|T| \geq 3$, and in time $2^k \cdot n^{\mathcal{O}(1)}$, either returns that $\mathcal{F}(T, k, G) = \emptyset$, or returns a strong separator S corresponding to the instance, and q components $C_1, C_2, \ldots, C_q$ satisfying the property stated in Definition 15.*

A Recursive Algorithm for GIMC. We design a recursive algorithm called `IMWCut` for the problem Generalized Independent Multiway Cut (GIMC) that takes the following as input.

- A graph $G = (V, E)$,
- a set $T \subseteq V(G)$ of terminals,
- a set $Q \subseteq V(G)$ of special vertices, with $T \subseteq Q$,
- Matroid $\mathcal{M} = (U, \mathcal{I})$ of rank r, where $V(G) \setminus Q \subseteq U$, and
- two non-negative integers k, q where $k + q \leq r$.

Given an instance, first if $|T| = 2$, then the problem is equivalent to IVstC, and can be directly solved using Theorem 4. Otherwise, suppose that $|T| \geq 3$. Then, first in time $\mathcal{O}^*(2^k)$, we either compute a strong separator using Lemma 16, or conclude that we have a no-instance. If we find a strong separator $S \subseteq V$, then its definition lends itself to the following natural recursive strategy.

First, we handle solutions that have a non-empty intersection with S by branching on each vertex in S, and recursively calling the algorithm on the resulting instance with a smaller parameter. Each such recursive call returns a representative family of solutions containing the corresponding vertex of S. Next, we handle the solutions O that may intersect one of the q terminal components $C_1, C_2, \ldots, C_q$ with the intersection size being between 1 and $k-1$. Let us consider any such C_t, $t \in [q]$. We design a divide-and-conquer strategy to find representative family for "solutions that have a certain kind of structural interaction between C_t. For a solution O intersecting C_t, it holds that $1 \leq |O \cap C_t| \leq k-1$. Thus, for appropriately defined two instances of GIMC, we make two recursive calls to the same algorithm with

strictly smaller parameters. Each recursive call inductively returns a representative family for solutions that locally have the same separation properties as O. Then, we combine such families, and we do this for each set of $k^{\mathcal{O}(k)}$ guesses. Finally, we take the union of all solutions found over all recursive calls (solutions obtained by divide-and-conquer strategy are first appropriately combined), and then we compute the representative family of the resultant. This set is returned by the algorithm as the required $\mathcal{F}'(T,k,G) \subseteq^{r-k}_{\text{rep}} \mathcal{F}(T,k,G)$. A full description and the proof of correctness can be found in the full version [2]. We conclude here with the following theorem.

▶ **Theorem 17.** *There exists a deterministic algorithm for* Generalized Independent Multiway Cut *that runs in time* $f(r,k) \cdot n^{\mathcal{O}(1)}$*, and outputs* $\mathcal{F}'(T,k,G) \subseteq^{q}_{rep} \mathcal{F}(T,k,G)$ *such that* $|\mathcal{F}'(T,k,G)| = 2^{k+q}$*. Here* $\mathcal{F}(T,k,G)$ *denotes the set of all* k *size multiway cuts for* T *in* G*, assuming that there is no independent multiway cut of size* $k' < k$*.*

5 Conclusion

In this paper we studied matroidal generalizations of several fundamental graph separation problems, such as Vertex (s,t)-Cut and Vertex Multiway Cut. We showed that these problems are FPT when parameterized by the size of the solution. These results were obtained by combining several recent and old techniques in the world of parameterized complexity. Our paper leaves several interesting questions open. These include obtaining $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time algorithm for all the problems studied in this paper. In addition, designing FPT algorithms for matroid versions of other problems such as Vertex Multi-cut, Directed Feedback Vertex Set remains an interesting research direction.

References

1. Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
2. Aritra Banik, Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, Satyabrata Jana, and Saket Saurabh. Cuts in graphs with matroid constraints, 2024. arXiv:2406.19134.
3. Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011. doi:10.1137/080733991.
4. M. Cygan, F. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
5. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
6. Marek Cygan, Fabrizio Grandoni, and Danny Hermelin. Tight kernel bounds for problems on graphs with small degeneracy. *ACM Trans. Algorithms*, 13(3):43:1–43:22, 2017. doi:10.1145/3108239.
7. Uriel Feige and Mohammad Mahdian. Finding small balanced separators. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 375–384. ACM, 2006. doi:10.1145/1132516.1132573.
8. F. V. Fomin, D. Lokshtanov, and S. Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *proceedings of SODA*, pages 142–151, 2014.
9. Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

10 Gagan Goel, Chinmay Karande, Pushkar Tripathi, and Lei Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 755–764. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.81.

11 MohammadTaghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. Local search algorithms for the red-blue median problem. *Algorithmica*, 63(4):795–814, 2012. doi:10.1007/S00453-011-9547-9.

12 Zhong Huang and Xueliang Li. Hardness results for rainbow disconnection of graphs. *CoRR*, abs/1811.11939, 2018. arXiv:1811.11939.

13 Stefanie Jegelka and Jeff Bilmes. Notes on graph cuts with submodular edge weights. In *NIPS 2009 Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity Polyhedra (DISCML)*, pages 1–6, 2009.

14 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Flow-augmentation i: Directed graphs. *arXiv preprint arXiv:2111.03450*, 2021.

15 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Solving hard cut problems via flow-augmentation. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 149–168. SIAM, 2021. doi:10.1137/1.9781611976465.11.

16 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Directed flow-augmentation. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 938–947. ACM, 2022. doi:10.1145/3519935.3520018.

17 Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 646–659. ACM, 2018. doi:10.1145/3188745.3188882.

18 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. *ACM Trans. Algorithms*, 14(2):14:1–14:20, 2018. doi:10.1145/3170444.

19 László Lovász. Flats in matroids and geometric graphs. In *Combinatorial Surveys (Proc. 6th British Combinatorial Conference*, pages 45–86, 1977.

20 László Lovász. *Graphs and geometry*, volume 65 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, 2019. doi:10.1090/coll/065.

21 Pasin Manurangsi, Erel Segal-Halevi, and Warut Suksompong. On maximum bipartite matching with separation. *Inf. Process. Lett.*, 182:106388, 2023. doi:10.1016/J.IPL.2023.106388.

22 Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/J.TCS.2005.10.007.

23 Dániel Marx. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.*, 410(44):4471–4479, 2009. doi:10.1016/j.tcs.2009.07.027.

24 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Treewidth reduction for constrained separation and bipartization problems. In Jean-Yves Marion and Thomas Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*, volume 5 of *LIPIcs*, pages 561–572. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPICS.STACS.2010.2485.

25 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Trans. Algorithms*, 9(4):30:1–30:35, 2013. doi:10.1145/2500119.

26 Syed Mohammad Meesum, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Rank vertex cover as a natural problem for algebraic compression. *SIAM J. Discret. Math.*, 33(3):1277–1296, 2019. doi:10.1137/17M1154370.

27 Rolf Niedermeier and Peter Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. *J. Algorithms*, 47(2):63–77, 2003. doi:10.1016/S0196-6774(03)00005-1.

28 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/J.ORL.2003.10.009.

29 Hadas Shachnai and Meirav Zehavi. A multivariate framework for weighted FPT algorithms. *J. Comput. Syst. Sci.*, 89:157–189, 2017. doi:10.1016/J.JCSS.2017.05.003.

30 Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.